The Chombo Framework for Block-Structured Adaptive Mesh Refinement Algorithms

Dan Martin
Applied Numerical Algorithms Group (ANAG)
Lawrence Berkeley National Laboratory

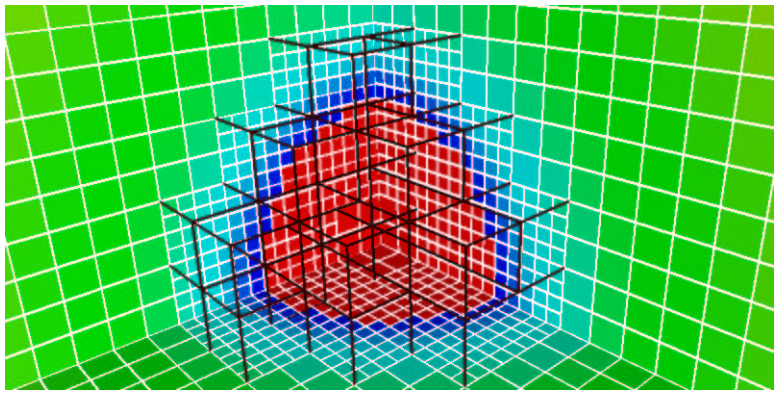
Local Refinement for Partial Differential Equations

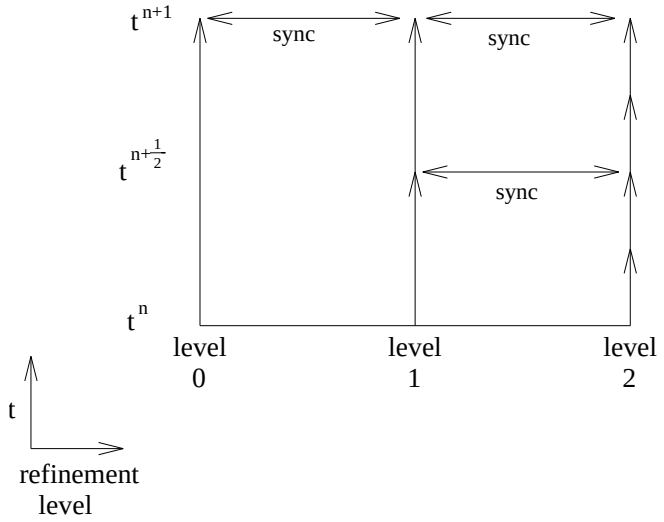
Variety of problems that exhibit multiscale behavior, in the form of localized large gradients separated by large regions where the solution is smooth.

- Shocks and interfaces.
- Self-gravitating flows in astrophysics.
- Complex engineering geometries.
- Combustion.
- Magnetohydrodynamics: space weather, magnetic fusion.

In adaptive methods, one adjusts the computational effort locally to maintain a uniform level of accuracy throughout the problem domain.

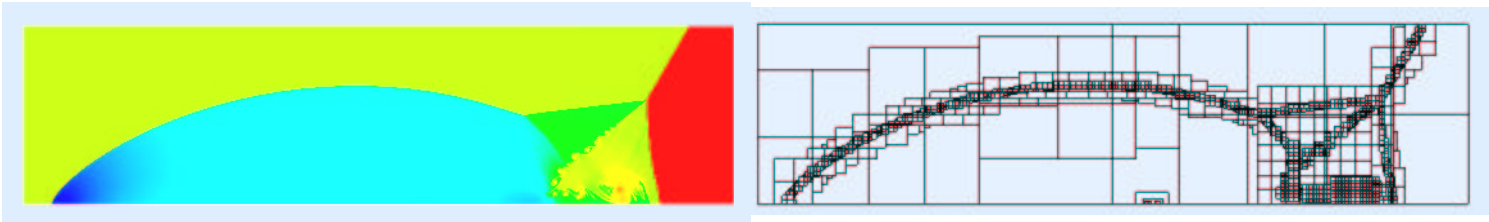
Block-Structured Local Refinement (Berger and Oliger, 1984)





Refined regions are organized into logically rectangular patches

Refinement performed in time as well as in space.



Chombo: a Software Framework for Block-Structured AMR

Requirement: to support a wide variety of applications that use block-structured AMR using a common software framework.

• Mixed-language model: C++ for higher-level data structures, Fortran for regular single-grid calculations.

• Reuseable components. Component design based on mapping of mathematical abstractions to classes.

• Build on public-domain standards: MPI, HDF5, VTK.

• Interoperability with other SciDAC ISIC tools: grid generation (TSTT), solvers (TOPS), performance analysis tools (PERC).

Previous work: BoxLib (LBNL/CCSE), KeLP (Baden, et. al., UCSD), FIDIL (Hilfinger and Colella).

Layered Design

• **Layer 1.** Data and operations on unions of boxes – set calculus, rectangular array library (with interface to Fortran), data on unions of rectangles, with SPMD parallelism implemented by distributing boxes over processors.

• **Layer 2.** Tools for managing interactions between different levels of refinement in an AMR calculation – interpolation, averaging operators, coarse-fine boundary conditions.

• **Layer 3.** Solver libraries – AMR-multigrid solvers, Berger-Oliger time-stepping.

• **Layer 4.** Complete parallel applications.

• **Utility layer.** Support, interoperability libraries – API for HDF5 I/O, visualization package implemented on top of VTK, C API's.

Examples of Layer 1 Classes (`BoxTools`)

• `IntVect` $\boldsymbol{i} \in \mathbb{Z}^d$. Can translate $\boldsymbol{i}_1 \pm \boldsymbol{i}_2$, coarsen $\frac{\boldsymbol{i}}{s}$, refine $\boldsymbol{i} * s$.

• `Box` $B \subset \mathbb{Z}^d$ is a rectangle: $B = [\boldsymbol{i}_{low}, \boldsymbol{i}_{high}]$. B can be translated, coarsened, refined. Supports different centerings (node-centered vs. cell-centered) in each coordinate direction.

• `IntVectSet` $\mathcal{I} \subset \mathbb{Z}^d$ is an arbitrary subset of $\mathbb{Z}^d$. $\mathcal{I}$ can be shifted, coarsened, refined. One can take unions and intersections, with other `IntVectSet`s and with `Box`es, and iterate over an `IntVectSet`. Useful for representing irregular sets.

• `FArrayBox A(Box B, int nComps)`: multidimensional arrays of Reals constructed with `B` specifying the range of indices in space, `nComp` the number of components. `Real* FArrayBox::dataPointer` returns pointer to the contiguous block of data that can be passed to Fortran.

Example: explicit heat equation solver on a single grid

```
// C++ code:

  Box domain(IntVect:Zero,(nx-1)*IntVect:Unit);
  FArrayBox soln(grow(domain,1), 1);
  soln.setVal(1.0);

  for (int nstep = 0;nstep < 100; nstep++)
 {
  heatsub2d_(soln.dataPtr(0),
           &(soln.loVect()[0]), &(soln.hiVect()[0]),
           &(soln.loVect()[1]), &(soln.hiVect()[1]),
             domain.loVect(), domain.hiVect(),
             &dt, &dx, &nu);
 }
```

```
c Fortran code:
      subroutine heatsub2d(phi,nlphi0, nhphi0,nlphi1, nhphi1,
     &     nlreg, nhreg, dt, dx, nu)

      real*8 lphi(nlphi0:nhphi0,nlphi1:nhphi1)
      real*8  phi(nlphi0:nhphi0,nlphi1:nhphi1)
      real*8 dt,dx,nu
      integer nlreg(2),nhreg(2)

c Remaining declarations, setting of boundary conditions goes here.
...
       do j = nlreg(2), nhreg(2)
         do i = nlreg(1), nhreg(1)
            lapphi =
     &           (phi(i+1,j)+phi(i,j+1)
     &           +phi(i-1,j)+phi(i,j-1)
     &           -4.0d0*phi(i,j))/(dx*dx)

            lphi(i,j) = lapphi
         enddo
       enddo
```

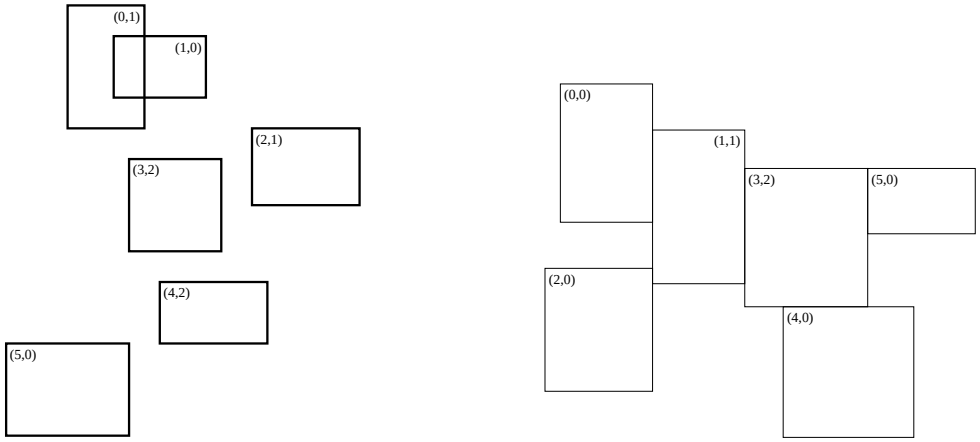
```
c Increment solution with rhs.

      do j = nlreg(2), nhreg(2)
         do i = nlreg(1), nhreg(1)
            phi(i,j) = phi(i,j) + nu*dt*lphi(i,j)
         enddo
      enddo

      return
      end
```

Distributed Data on Unions of Rectangles

Provides a general mechanism for distributing data defined on unions of rectangles onto processors, and communications between processors.



• Metadata of which all processors have a copy: `BoxLayout` is a collection of `Box`es and processor assignments: $\{B_k, p_k\}_{k=1}^{nGrids}$. `DisjointBoxLayout:public BoxLayout` is a `BoxLayout` for which the `Box`es must be disjoint.

• `template <class T> LevelData<T>` and other container classes hold data distributed over multiple processors. For each `k = 1 ... nGrids`, an "array" of type `T` corresponding to the box B_k is allocated on processor p_k. Straightforward API's for copying, exchanging ghost cell data, iterating over the arrays on your processor in a SPMD manner.

Software Reuse by Templating Dataholders

Classes can be parameterized by types, using the class template language feature in C++.

`BaseFAB<T>` is a multidimensional array for any type `T`.

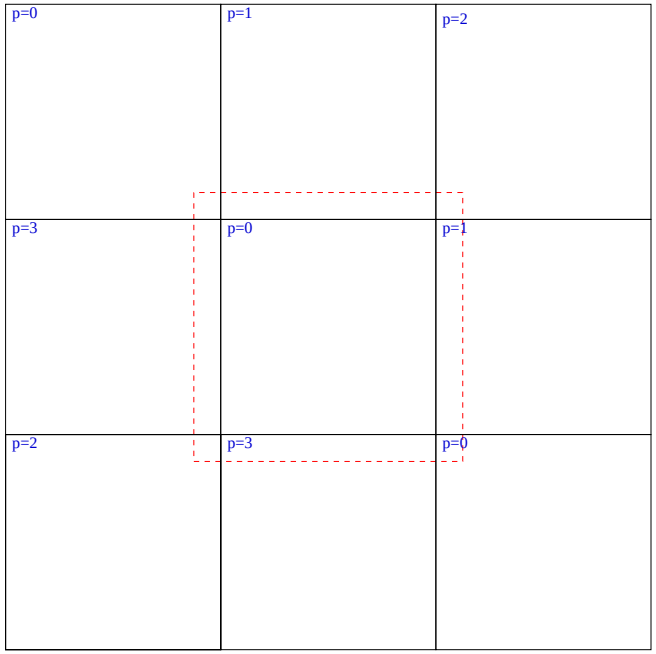
```
FArrayBox:  public BaseFAB<Real>
```

In `LevelData<T>`, `T` can be any type that "looks like" a multidimensional array.

Examples include:

- Ordinary multidimensional arrays, e.g. `LevelData<FArrayBox>`.
- A composite array type for supporting embedded boundary computations
- Binsorted lists of particles, e.g. `BaseFab<List<ParticleType>>`

Example: explicit heat equation solver, parallel case



Want to apply the same algorithm as before, except that the data for the domain is decomposed into pieces and distributed to processors.

• `LevelData<T>::exchange()`: obtains ghost cell data from valid regions on other patches.

• `DataIterator`: iterates over only the patches that are owned on the current processor.

```
// C++ code:
  Box domain(IntVect:Zero,(nx-1)*IntVect:Unit);
  DisjointBoxLayout dbl;
// Break domain into blocks, and construct the DisjointBoxLayout.
  makeGrids(domain,dbl,nx);

  LevelData<FArrayBox> phi(dbl, 1, IntVect::TheUnitVector());

  for (int nstep = 0;nstep < 100;nstep++)
  {
...
// Apply one time step of explicit heat solver: fill ghost cell value
// and apply the operator to data on each of the Boxes owned by this
// processor.

  phi.exchange();
  DataIterator dit = dbl.dataIterator();

// Iterator iterates only over those boxes that are on this processor
```

```
for (dit.reset();dit.ok();++dit)
{
FArrayBox& soln = phi[dit()];
Box& region = dbl[dit()];
heatsub2d_(soln.dataPtr(0),
         &(soln.loVect()[0]), &(soln.hiVect()[0]),
         &(soln.loVect()[1]), &(soln.hiVect()[1]),
           region.loVect(), region.hiVect(),
           domain.loVect(), domain.hiVect(),
           &dt, &dx, &nu);
}
}
```

Layer 2: Coarse-Fine Interactions (`AMRTools`).
The operations that couple different levels of refinement are among the most difficult to implement AMR.

- Interpolating between levels (`FineInterp`).
- Averaging down onto coarser grids (`CoarseAverage`).
- Interpolation of boundary conditions (`PWLFillpatch, QuadCFInterp`).
- Managing conservation at coarse-fine boundaries (`LevelFluxRegister`).

These operations typically involve interprocessor communication and irregular computation.

Layer 3: Reusing Control Structures Via Inheritance (`AMRTimeDependent`, `AMRElliptic`).
AMR has multilevel control structures which are largely independent of the details of the operators and the data.

• Berger-Oliger timestepping (refinement in time).

• Multigrid iteration on a union of rectangles. (single AMR level)

• Multigrid iteration on an AMR hierarchy. (multilevel AMR solve)

To separate the control structure from the details of the operations that are being controlled, we use C++ inheritance in the form of *interface classes*.

Layer 4: AMR Applications

• A general driver for an unsplit second-order Godunov method for hyperbolic conservation laws. User provides physics-dependent components (characteristic analysis, Riemann solver).

• Level solvers, AMR multigrid solvers for Poisson, Helmholtz equations.

• Incompressible Navier-Stokes solver using projection method. Includes projection operators for single level, AMR hierarchy. Advection-diffusion solvers.

• Wave equation solver.

• Time-dependent Ginzburg-Landau equation solver.

• Volume-of-fluid algorithm fluid-solid interactions.

AMR Utility Layer

• API for HDF5 I/O.

• Interoperability tools. We are developing a framework-neutral representation for pointers to AMR data, using opaque handles. This will allow us to wrap Chombo classes with a C interface and call them from other AMR applications.

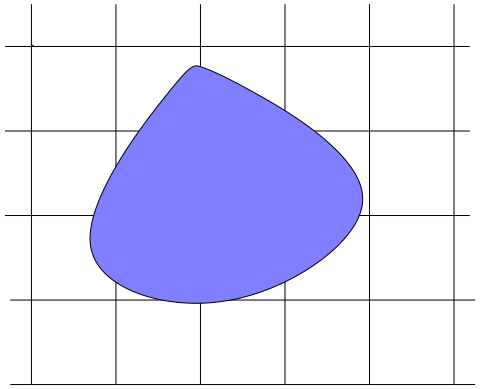
• Chombo Fortran - a macro package for writing dimension-independent Fortran and managing the Fortran / C interface.

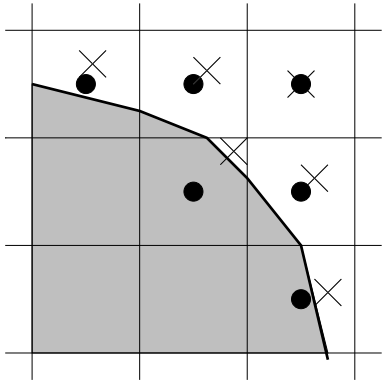
• `ParmParse` class from BoxLib for handling input files.

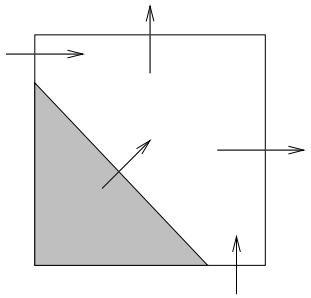
• Visualization and analysis tools (ChomboVis).

Cartesian Grid Representation of Irregular Boundaries (`EBChombo`)

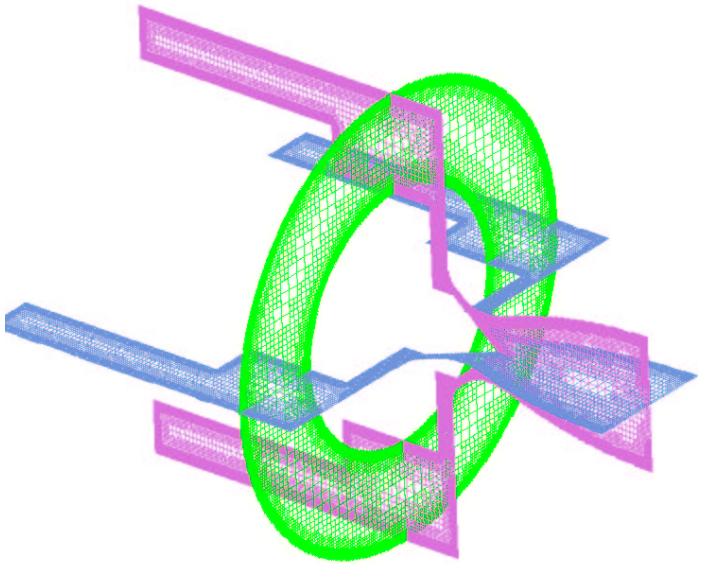
Based on nodal-point representation (Shortley and Weller, 1938) or finite-volume representation (Noh, 1964).

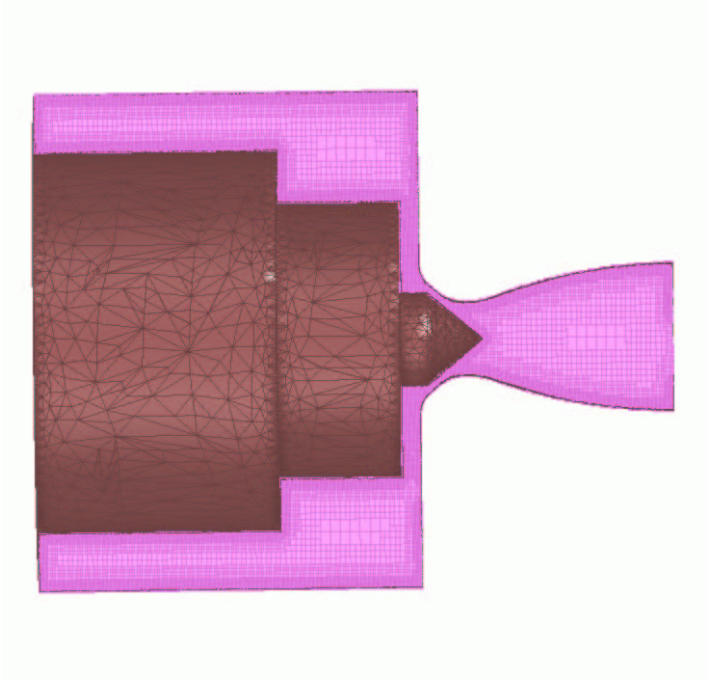


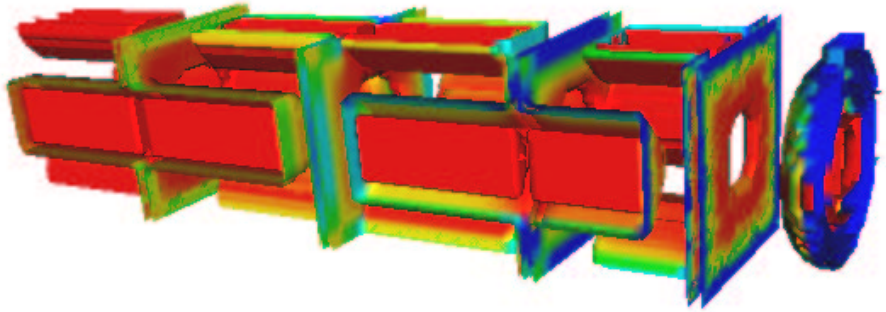


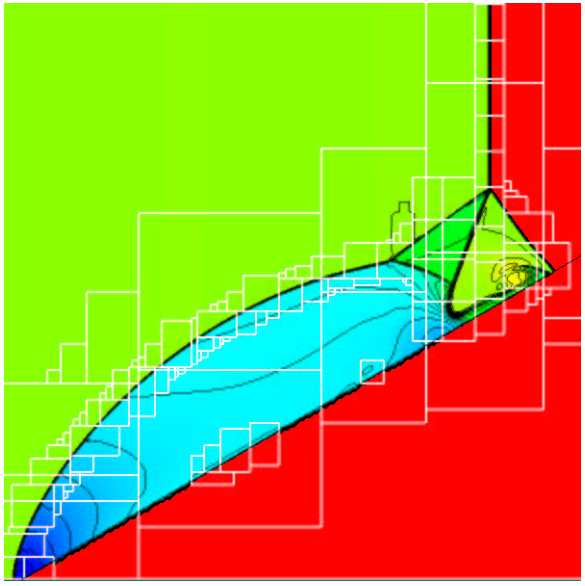


Computation in Complex Geometry









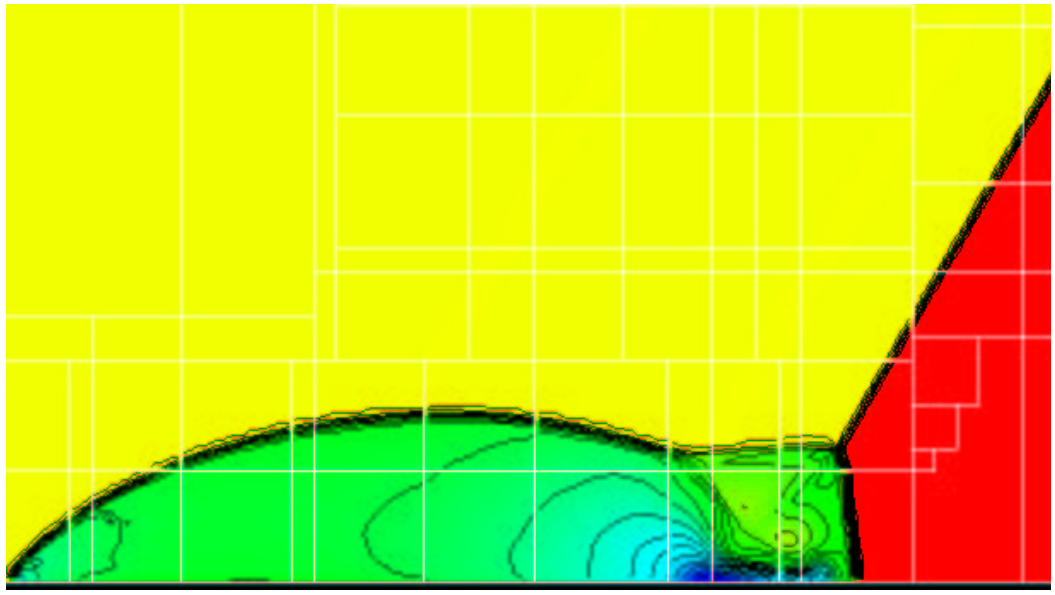
Current Applications (partial list)

- Accelerator design and modelling (LBNL)
- Magnetic fusion (R. Samtaney, S. Jardin, PPPL)
- Star formation; multiphase microgravity flows (NASA CT Program).
- Solid mechanics (G. Miller, UC Davis and LBNL).
- Time-dependent Ginzburg-Landau equations (F. Alexander, LANL).
- Semi-local strings (J. Borrill, LBNL).
- Cosmology (F. Miniati, MPI-Garching).
- Low-Mach number geophysical, astrophysical flows (UC Davis, Univ. of Chicago ASCI Center).

Gas Dynamics

Unsplit higher-order Godunov scheme with AMR for hyperbolic systems of equations (AMRGodunov).

Includes flux correction at coarse-fine interfaces for conservation.



Elliptic Equations

AMR elliptic solver – used as standalone code (AMRPoisson) or as solver library (AMRElliptic: AMRSolver class)

- Implements a multigrid solver for an AMR hierarchy of refined grids.
- Uses multilevel discretizations of the elliptic operators to maintain accuracy in the presence of coarse-fine interfaces.

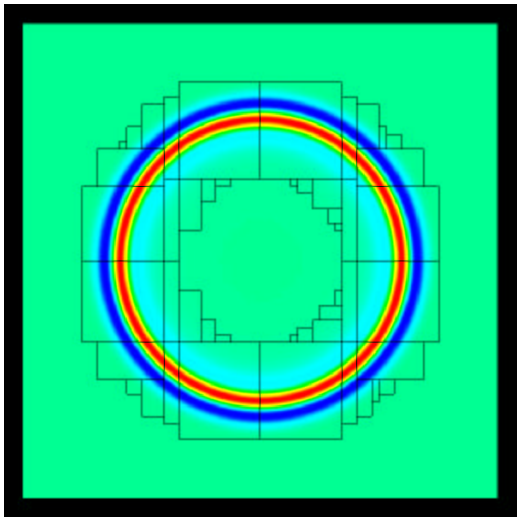
Wave Equation Solver

• Write second-order wave equation as first-order system in time.

$$\frac{\partial \varphi}{\partial t} = \pi$$
$$\frac{\partial \pi}{\partial t} = \Delta \varphi$$

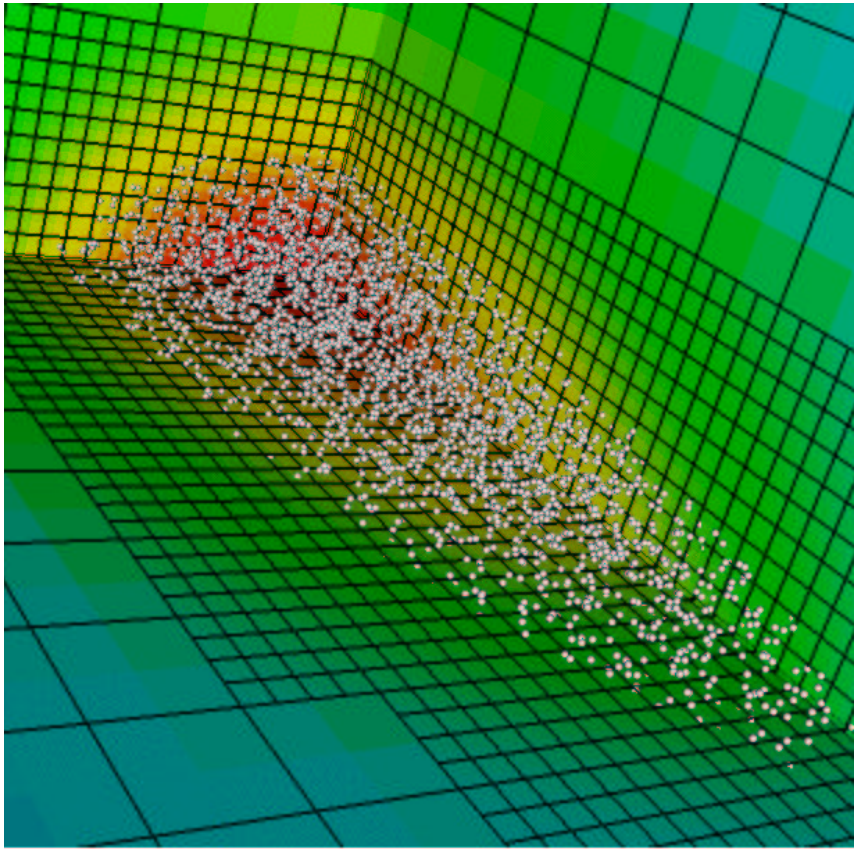
• Discretize Laplacian on AMR grid using RK4 in time, quadratic interpolation in space for coarse-fine boundary conditions.

• Refinement in time: linear interpolation in time for coarse-fine boundary conditions, treat π as a conserved quantity for the purpose of refluxing.



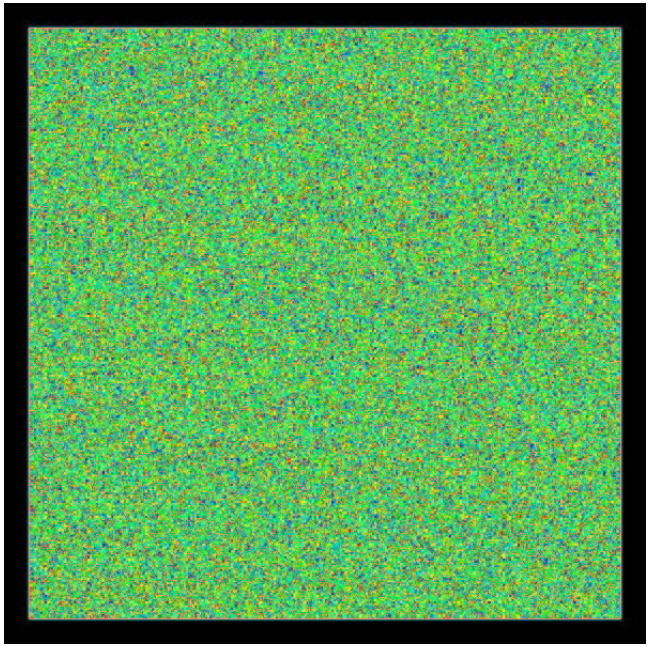
Particles

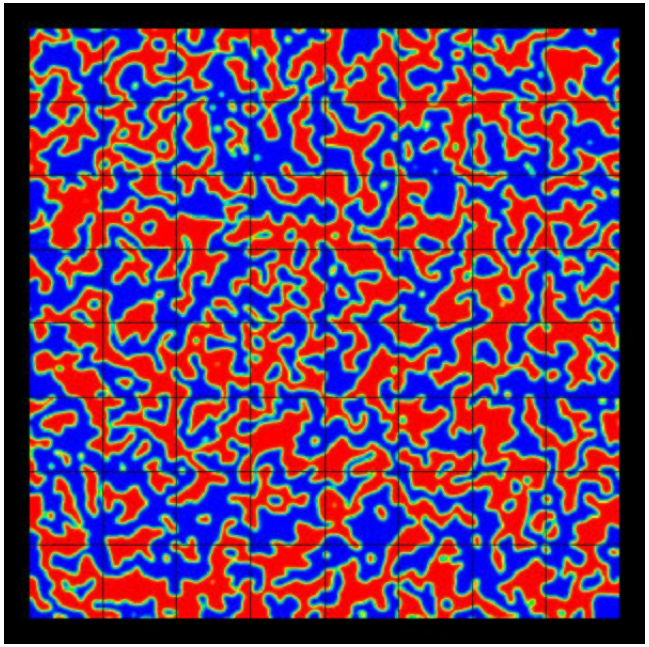
AMR-PIC code used in accelerator modelling

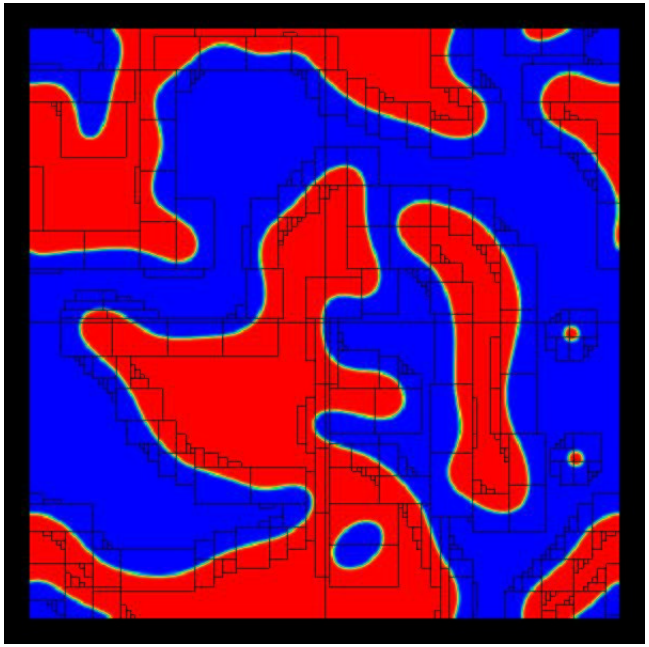


Time-dependent Ginzberg-Landau Equations

Diffusion equation with a nonlinear source term – used to model phase-field dynamics (crystal growth, etc) (with F. Alexander, LANL)



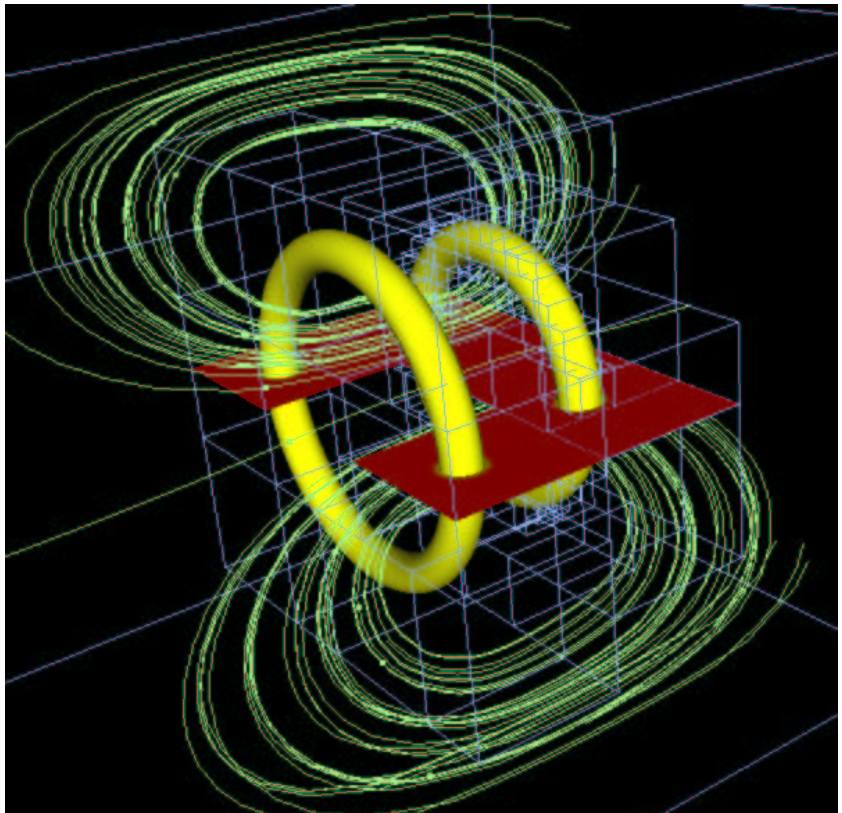




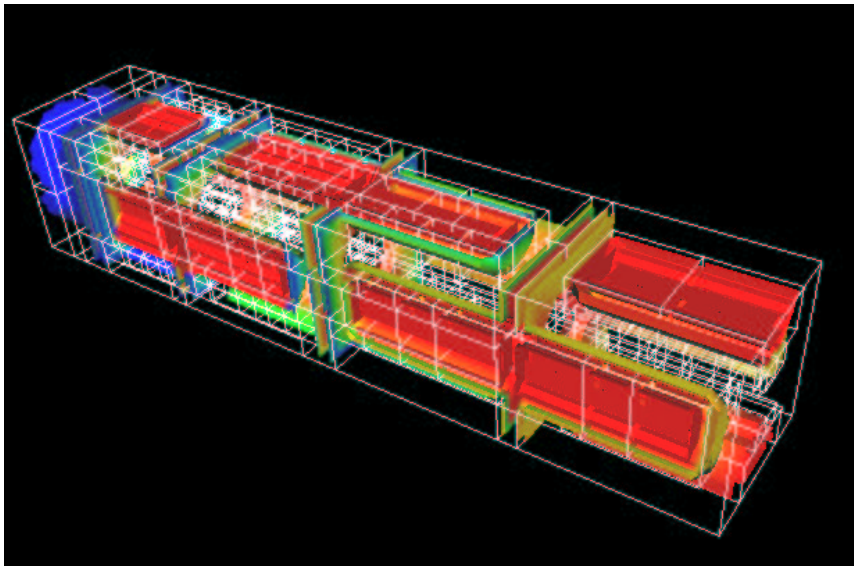
Incompressible AMR Navier-Stokes

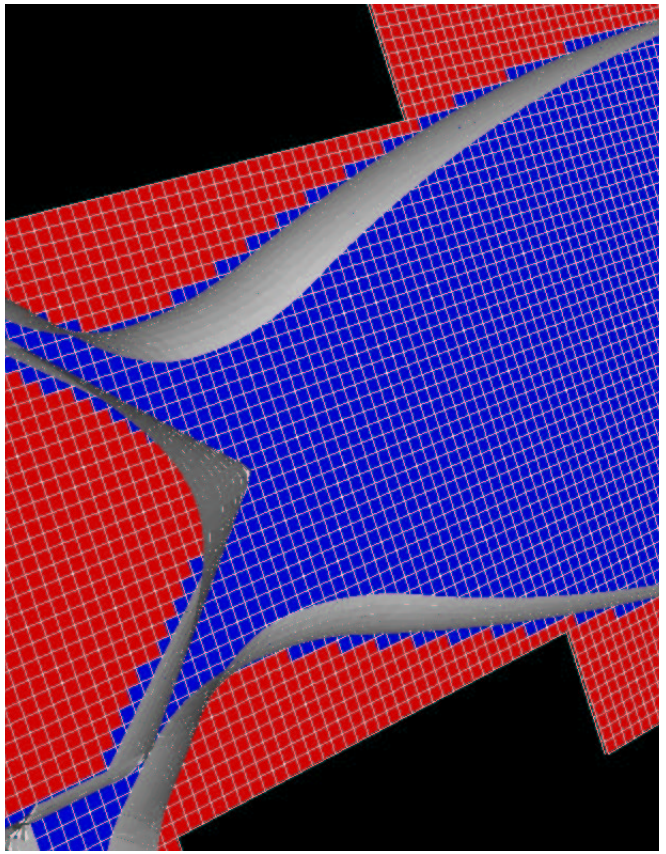
AMRINS code

- Implements a projection method for incompressible viscous flow.
- Freestream preservation maintained approximately in the presence of coarse-fine interfaces using an advection velocity correction computed using an auxiliary advected scalar.
- Viscous updates performed using L_0-stable semi-implicit Runga-Kutta scheme



Accelerator Design

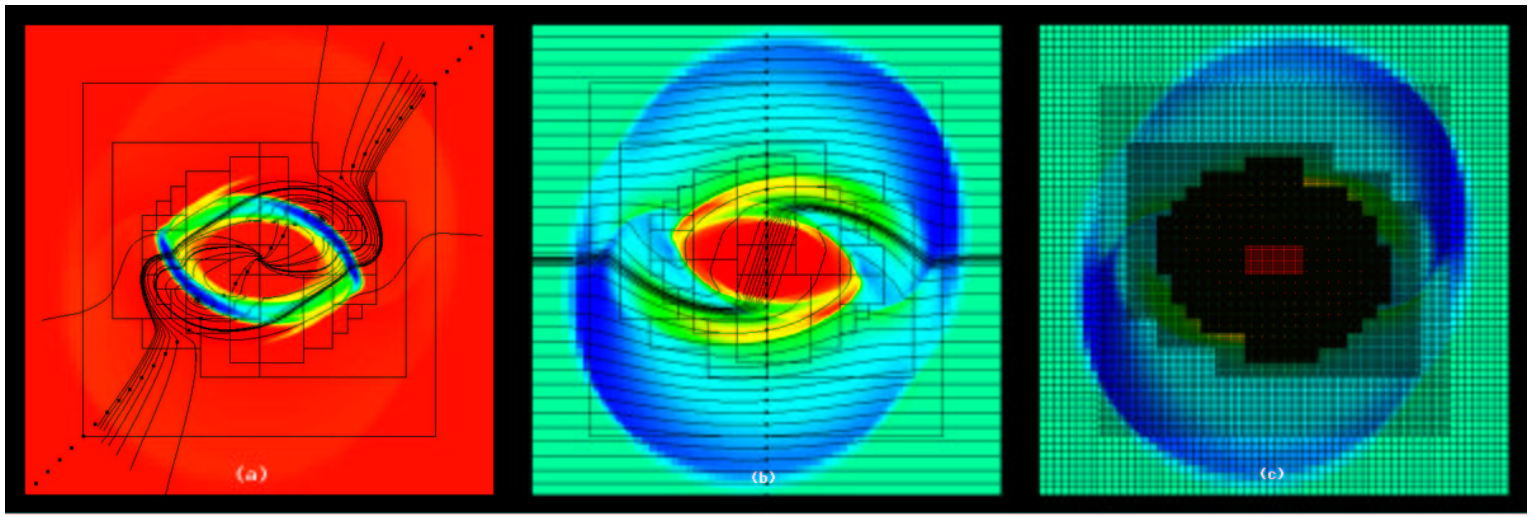




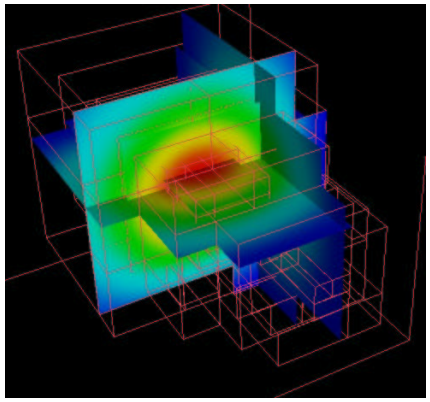
Magnetohydrodynamics (Samtaney, et. al., 2003)

Fluid representation: AMR for magnetohydrodynamics, based on semi-implicit methods.

- Explicit upwind discretizations for hyperbolic terms.
- Implicit discretizations for parabolic operators.
- Projection to enforce $\nabla \cdot \vec{B} = 0$ constraint.



ChomboVis Interactive Visualization and Analysis Tools



• "AMR-aware"

– Block-structured representation of the data leads to efficiency.

– Useful as a debugging tool (callable from debuggers (gdb))

• Visualization tools based on VTK, a open-source visualization library.

• Implementation in C++ and Python

– GUI interface for interactive visualization

– Command-line python interface to visualization and analysis tools, batch processing capability – goal is a full analysis tool.

• Interface to HDF5 I/O along with C API provides access to broad range of AMR users. ("Framework-neutral")

Where it's heading

Goal is for ChomboVis to become a full data analysis and visualization tool.

- Further evolution of Python scripting interface
- Data analysis functionality
- "Out of core" and parallel functionality for large datasets
- non-rectilinear coordinate systems
- Presentation graphics

Chombo, ChomboVis available from the ANAG website:

- http://seesar.lbl.gov/ANAG/software.html
- New Release, August 15, 2003.

Acknowledgements
DOE Applied Mathematical Sciences Program
DOE HPCC Program
DOE SciDAC Program
NASA Earth and Space Sciences Computational Technologies Program